

ECE 6970: Real-Time Systems Homework 2

Due on March 13, 2007

(1) Consider the following task set (σ_i is the set of critical sections that task T_i will need at some point of its execution):

Task	e_i	P_i	σ_i
T_1	4	10	S_1, S_2
T_2	5	15	S_2, S_3
T_3	10	35	S_3
T_4	1	36	S_1

Critical Section	Execution Time
S_1	2
S_2	1
S_3	2

Is this task set RM-schedulable under the priority ceiling algorithm?

(2) Show that the Priority Ceiling algorithm is not optimal. Do this by creating a task set which has unnecessary blocking under Priority Ceiling, i.e., if we removed the blocking, the system would still operate correctly without any deadlocks.

(3) Is the following task set EDF-schedulable? Note that the deadlines are not equal to their respective periods.

Task	e_i	d_i	P_i
T_1	1	5	5
T_2	2	4	10
T_3	3	10	15

(4) In this question, we will consider the relative effectiveness of the first-fit and the utilization balancing approaches to task assignment. We proceed by generating task sets which we know to be schedulable and then check what fraction of them can be feasibly assigned using each of the above task assignment heuristics.

To generate task sets that can be feasibly allocated and scheduled on a set of n processors, where n is an input, start by constructing n task sets, each with total

processor utilization no greater than 1. This guarantees that at least one assignment exists, under which EDF can meet all deadlines. Then, take the union of all these task sets and present this union to the task-assignment heuristics. Write software to generate sets of m tasks each, with total utilization u . Both m and u are inputs to the program. The utilization of each task should be chosen randomly with the proviso that the sum of the utilizations in any given set is equal to u .

Now, generate the union of all these task sets. You'll have a total of nm tasks. Do the following:

- (a) Sort the tasks in ascending order of their utilizations. Present this list of tasks to a first-fit allocation algorithm and check if it is able to successfully allocate them.
- (b) Sort the tasks in descending order of their utilizations. Present this list of tasks to a first-fit allocation algorithm and check if it is able to successfully allocate them.
- (c) Repeat (a) for the utilization-balancing heuristic.
- (d) Repeat (b) for the utilization-balancing heuristic.

For each set of parameters (specified below), compute the fraction, f , of instances that can be scheduled successfully. Also generate the 95% confidence intervals: the half-width, h , of these intervals is equal to $1.96\sqrt{f(1-f)/r}$, where r is the number of runs for that set of parameters. The 95% confidence interval is $[f - h, f + h]$. You should make enough runs to ensure that $h \leq 0.05f$.

Plot your results, with the average utilization per processor varying from 0.5 to 1.0 on the x-axis and the success rate, f , on the y axis. (If the curve is flat for most of that interval, you may want to provide additional plots zooming in on the "interesting" regions). Assume that you have a total of $n = 4$ processors and plot curves for $m = 3, 5, 7, 9$ (i.e., the total number of tasks will be 12, 20, 28, 36).

Investigate the impact of varying the number of processors, n . Plot results, with n on the x-axis and f on the y-axis; varying n from 2 to 10, in steps of 1. Plot curves for the following average utilizations per processor: 0.9, 0.95, 0.975. Assume $3n$ tasks in all.

Note: Due to the programming problem, this homework carries twice the weight of a "usual" homework.